观察者模式

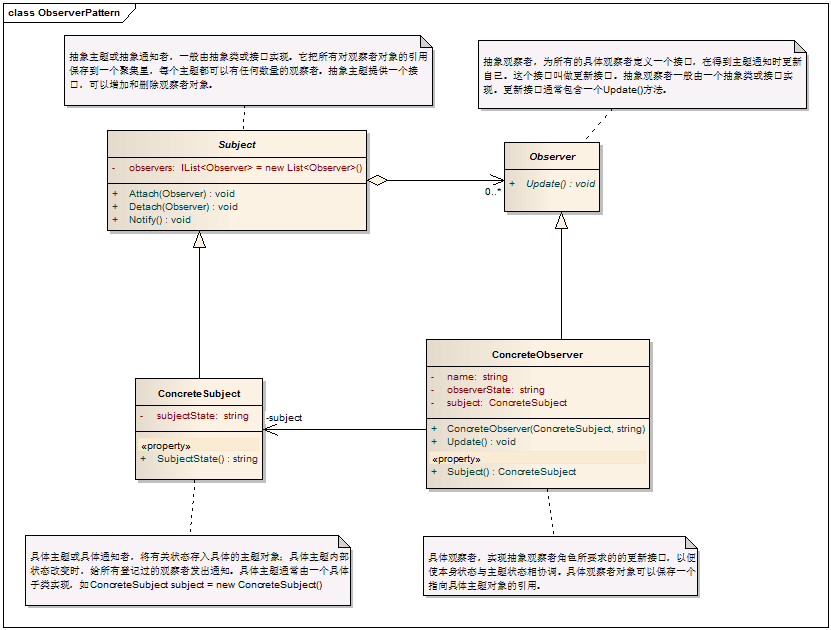
一、观察者模式

有时被称作发布/订阅模式，观察者模式定义了一种一对多的依赖关系，让多个观察者对象同时监听某一个主题对象。这个主题对象在状态发生变化时，会通知所有观察者对象，使它们能够自动更新自己。将一个系统分割成一个一些类相互协作的类有一个不好的副作用，那就是需要维护相关对象间的一致性。我们不希望为了维持一致性而使各类紧密耦合，这样会给维护、扩展和重用都带来不便。观察者就是解决这类的耦合关系的

1. Gitub地址

*https://github.com/guigarage/ObserverPattern/blob/master/api/src/main/java/javax/observer/Observable.java*

三、观察者模式类图



1. 代码

*package javax.observer;*

*import java.util.Optional;*

*/\*\**

*\* This interfaces provides general support of the observer pattern. Implementations / Instances might*

*\* work as a wrapper around a basic variable and will fire change events whenever the value of the variable*

*\* has changed.*

*\* @author Hendrik Ebbers*

*\*/*

*public interface Observable<V> {*

*/\*\**

*\* Returns the current value.*

*\* @return the value*

*\*/*

*V getValue();*

*/\*\**

*\* Returns an {@link Optional} that contains the current value or {@code null}. This is a*

*\* convenience method that can be used to acces the value by using a fluent API based on*

*\* the {@link Optional} API.*

*\* @return an {@link Optional} that contains the current value or {@code null}*

*\*/*

*default Optional<V> value() {*

*return Optional.ofNullable(getValue());*

*}*

*Subscription onChanged(ValueChangedListener<? super V> listener);*

*default Subscription onChangedAndCall(final ValueChangedListener<? super V> listener) {*

*final Subscription subscription = onChanged(listener);*

*listener.valueChanged(new ValueChangedEvent<V>() {*

*@Override*

*public Observable<V> getObservable() {*

*return Observable.this;*

*}*

*@Override*

*public V getValue() {*

*return Observable.this.getValue();*

*}*

*});*

*return subscription;*

*}*

*Subscription onWillChange(ValueWillChangeListener<? super V> listener);*

*}*

1. 代码分析

1、基于subjectd的要求，需要首先实现一个基本的Subject接口；

2、创建一个相关联的观察者。它需要有一个方法能使Subject附属于一个观察者；

3、关联已经建立接着实现具体的主题，他们将一直关注subject对象；

4、MySubject类就是我们的主对象，Observer是依赖于MySubject的对象，当MySubject变化时，Observer必然变化。AbstractSubject类中定义着需要监控的对象列表，可以对其进行修改：增加或删除被监控对象，且当MySubject变化时，负责通知在列表内存在的对象，observer就是一个接口。

**·**被观察者：这个角色可以是接口，也可以是抽象类或者具体的类，因为很多情况下会与其他的模式混用，所以使用抽象类的情况比较多。  
 **·**观察者：观察者角色一般是一个接口，它只有一个update方法，在被观察者状态发生变化时，这个方法就会被触发调用。  
 **·**具体的被观察者：使用这个角色是为了便于扩展，可以在此角色中定义具体的业务逻辑。观察者接口的具体实现，在这个角色中，将定义被观察者对象状态发生变化时所要处理的逻辑。

六、总结

通过依赖抽象而不是依赖具体类，去实现一个类中某个状态的改变，而通知相关的一些类去做出相应的改变，进而保持同步状态。实现这样的方式或许有很多种，但是为了使系统能够易于复用，应该选择第耦合度的方案。减少对象之间的耦合度有利于系统的复用，在保证低耦合度的前提下并且能够维持行动的协调一致，保证高度协作，观察者模式是一种很好的设计方案。